Robust Spacecraft Sequencing through Model-Based Programming

Michel D. Ingham & Brian C. Williams

Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

http://www.ai.mit.edu

The Problem: Robotic spacecraft for exploration of the solar system and beyond must be designed to navigate and operate in harsh, dynamic and uncertain environments. Such operating conditions make it difficult for the systems engineers and programmers responsible for encoding onboard software sequences to meet the requirements for very high reliability. We propose a robust solution to challenging sequencing problems (e.g. Planetary Entry, Descent and Landing), which combines discrete configuration commanding with continuous navigation and control, and will allow systems engineers to generate "executable specifications" of desired spacecraft behavior.

Motivation: Operational sequences for robotic spacecraft can be quite complex, involving multiple hardware reconfigurations, navigation computations and controlled maneuvers. Providing robustness for such spacecraft in the face of highly dynamic or uncertain environments requires the following: high-reliability software; fault protection built into the control sequence; a highly reactive sense-decide-act loop; and tight coupling between attitude/position control and spacecraft configuration control.

One particularly relevant illustration of the importance of robust sequence execution was the recent loss of the Mars Polar Lander spacecraft during Mars entry, descent and landing (EDL, see Figure 1). After months of analysis, the failure investigation team concluded that the vehicle most likely crashed into Mars because it incorrectly shut down its engine at 40 meters above the planetary surface. This failure resulted from a misinterpretation of the vehicle's dynamics, in this case due to a faulty software monitor. Using a traditional embedded software approach, it is difficult to anticipate low-level subsystem interactions and explicitly encode responses to each possible fault. We would like to support spacecraft systems engineers and software programmers with a new type of embedded language that operates directly on system states, allows reasoning from commonsense models of the system, and accommodates seamless integration of trajectory planning and control.

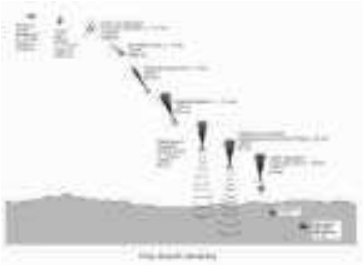

Figure 1: Entry, Descent and Landing Scenario

Previous Work: This effort develops the notion of model-based programming introduced in [3], and folds in results from previous work in model-based reasoning for discrete hardware reconfiguration [7], trajectory planning based on mathematical programming theory [4] and control theory. The modeling formalism leverages work in Timed Automata [1], from the Hybrid Systems field. This work builds off of the model-based execution capability developed for the Deep Space 1 spacecraft, and currently being extended for the Space Technology 7 mission. We also rely on ongoing work in Hybrid Mode Estimation [6].

Approach: Our approach is based on developing a new paradigm for embedded programming, called *model-based programming*. The underlying principle is that control programs can be written by asserting and checking states which may be "hidden", i.e. not directly controllable or observable, rather than by operating on observable and control variables. Such a control program is input to a sequencing engine, for onboard execution. An underlying model-based executive deduces the system state from observables and figures out how to achieve specified goal states. This model-based executive is comprised of:

- a hybrid mode estimation engine, which can infer both continuous states and discrete modes of the system [6].
- a reactive commanding engine, which takes in hardware configuration goals and reasons through a concurrent transition system model of the spacecraft to generate and execute an appropriate series of commands which will achieve the desired configuration. This deductive engine can reason about states of physical components in the system, or abstract states representing subsystem-level behavior.
- a navigation and control engine, which takes in continuous position and attitude goals, and generates appropriate maneuver commands to achieve the desired position/attitude. In its simplest form, this engine consists of some type of controller implementation (e.g. PID, LQ control). It can also include a trajectory planner to generate a sequence of maneuvers which will achieve the desired goal state (e.g. trajectory planning via linear programming techniques [4]).

In order to express model-based programs, a new embedded programming language has been developed, the Reactive Model-based Programming Language (RMPL) [3]. RMPL programs may be viewed as specifications of deterministic state transition systems, which act on the plant by asserting and checking constraints expressed in a propositional state logic. The propositions are assignments of state variables to values within their domains. Reactive combinators allow flexibility in expression of complex system behavior and dynamic relations. The constructs of RMPL are similar to those developed in TCC, a language for timed concurrent constraint programming [5].

Impact: Through the development of sequencer control programs using model-based programming, we will provide spacecraft systems engineers with the ability to directly encode executable specifications of the desired system behavior, e.g. as usually specified in State-Charts [2]. System robustness will be improved by coupling fault protection with navigation, control and discrete configuration commanding within the sequencer.

Future Work: To achieve the level of robustness discussed above, the current RMPL representation will be extended to accommodate assertion of discrete and continuous states, conditional branching on discrete states, continuous states and time, as well as clock initialization.

A Model-based Executive architecture will be established, which integrates the Hybrid Mode Estimation, Reactive Commanding, and navigation and control engines.

An important element of this work will be to study the interplay between trajectory planning, control, and fault protection, within the reactive sense-decide-act loop. The results of this study will shed significant light on the trades between system reactivity, robustness and versatility.

Research Support: This research is supported by NASA under contract NAG2-1388.

References:

[1] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[2] D. Harel. Statecharts: A visual approach to complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[3] M. Ingham, R. Ragno, and B. Williams. A reactive model-based programming language for robotic space explorers. In *Proceedings of i-SAIRAS 2001*, 2001.

[4] A. Richards, J. How, T. Schouwenaars, and E. Feron. Plume avoidance maneuver planning using mixed integer linear programming. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2001.

[5] V. Saraswat, R. Jagadeesan, and V. Gupta. Foundations of timed concurrent constraint programming. In *Proceedings of the Ninth Annual IEEE Symposium on Logic in Computer Science*, 1994.

[6] B. Williams, M. Hofbaur, and T. Jones. Mode estimation of probabilistic hybrid systems. Technical Report SSL-6-01, MIT Space Systems Laboratory, 2001.

[7] B. Williams and P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, 1996.